EFS Decryption: Unusual Circumstances

POLICE TECHNICAL
Technical. Training. Solutions.

David Willson, Sergeant with the Alaska Bureau of Investigation

Last year, the Alaska Bureau of Investigation began looking into allegations of Unlawful Exploitation of a Minor. A folder discovered during evidence examination had potential to completely change the case, but there was a problem: the crucial contents were encrypted.

Introduction

In early 2014, the Alaska Bureau of Investigation's Child Abuse Investigation Unit in Fairbanks began an investigation involving allegations of Unlawful Exploitation of a Minor, a felony sex crime under Alaska law.

Case Facts

A juvenile female alleged that a suspect took inappropriate photographs of her. She denied any hands-on offenses occurred. No further details were given.

The court granted a search warrant, authorizing the search of the suspect's residence, to include the seizure and subsequent forensic examination of computers and digital storage devices that might contain the alleged evidence of exploitation.

A search warrant was served and multiple computers, cellular phones, cameras, and loose storage devices were located.

An interview with the suspect resulted in an admission that he took photographs of the victim, but made no direct admissions regarding criminal behavior.

Initial Forensic Examination:
During the subsequent examinations of the evidence in the lab, hundreds of images of child sexual exploitation were found, supporting multiple charges of Possession of Child Pornography under Alaska law.

One folder, however, was titled with the victim's first name, and its contents were of particular interest in the investigation, potentially turning a Possession of Child Pornography case (mandatory sentencing range of 2-12 years) into a Sexual Abuse of a Minor case (mandatory sentencing range of 20-99 years). Unfortunately, the folder and its contents were encrypted with Microsoft's Encrypted File System (EFS).

Although a few of the abuse images were found to have been deleted on a separate volume and were subsequently recovered by AccessData's Forensic Tool Kit (FTK), the vast majority of the contents of the EFS-encrypted folder remained out of our reach.

Explanation of Microsoft's Encrypted File System

Under normal conditions, EFS use is relatively simple. The user of an EFS-enabled Windows system simply enables encryption by right-clicking the file or folder, then selecting Properties > Advanced > Encrypt contents to secure data (See Figure 1).

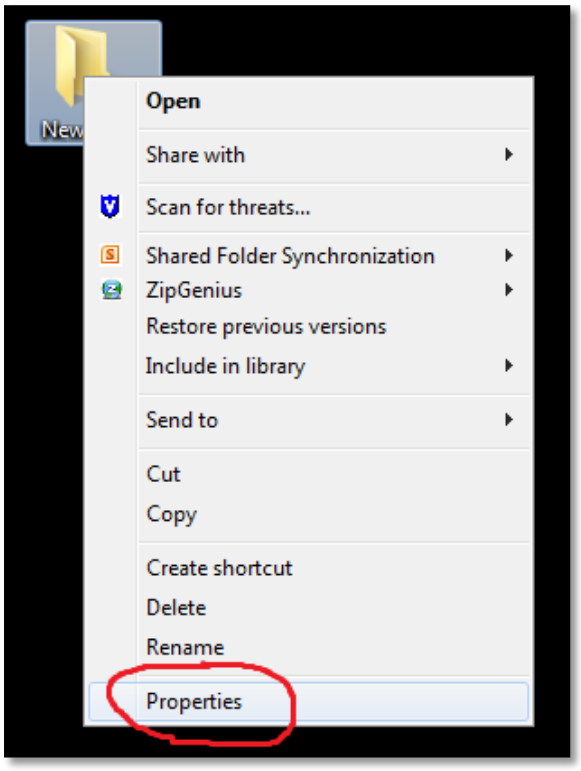

Figure 1: Enable encryption by right clicking on the file and selecting "Properties."

That file or folder then behaves as normal, allowing the user, when logged in, to transparently access the file(s) without further input; the files can be opened, moved, deleted, or modified like any other file on the volume. To decrypt, the user simply reverses the process, de-selecting the "Encrypt contents to secure data" box and all the files revert to their former state (See Figure 2).

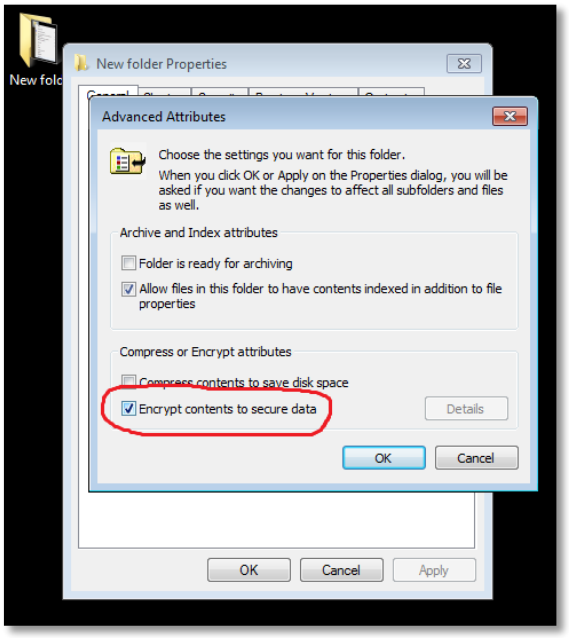
New folder Properties
Advanced Attributes
Choose the settings you want for this folder.
When you click OK or Apply on the Properties dialog, you will be asked if you want the changes to affect all subfolders and files as well.
Archive and Index attributes
Folder is ready for archiving
Allow files in this folder to have contents indexed in addition to file properties
Compress or Encrypt attributes
Compress contents to save disk space
Encrypt contents to secure data
Details
OK
Cancel
OK
Cancel
Apply

Figure 2: To decrypt a file, de-select "Encrypt contents to secure data."

When logging into the system as any other user, encrypted files would be unreadable to the user. The files are only readable to the user account that originally encrypted them and only on the Windows system that originally did the encryption.

The one exception to this is the use of a recovery certificate. The user, while logged in to the authorized windows account, can export a recovery certificate that may be later imported into another machine. Once the recovery certificate is installed on a separate machine, EFS encrypted files can be manipulated with full authority.

This is a simple process using a built in application, *certmgr.msc*, which is initiated from the command line to export/import the recovery certificate.

EFS is a file-system level encryption process and must remain on a volume that is formatted with NTFS to retain its attributes. If the authorized user moves those encrypted files to another NTFS volume, they will still retain their encryption and become inaccessible to any unauthorized users. If the authorized user moves those encrypted files to a FAT volume, however, Windows will decrypt those files immediately upon being copied.

Decrypting EFS under "normal" circumstances:

From a forensic perspective, automated decryption is possible under certain conditions. If the encrypted files are on the same volume where they were originally encrypted, then the credentials necessary for an examiner to decrypt them are often on that volume as well (located within the User/Application Data or User/Appdata folders, depending on the version of Windows).

The current version of Forensic ToolKit (v5.4) has an easy-to-use decryption tool that searches for encrypted files and their corresponding credentials (as long as they are on the same physical disk), and then decrypts the files. FTK won't work, however, if any of the following characteristics have been altered since the files were originally encrypted:

- SID (Windows Security Identifier – both domain and relative ID portions need to be identical to the original. See Figure 3)
- Windows login password
- Windows User Account Name

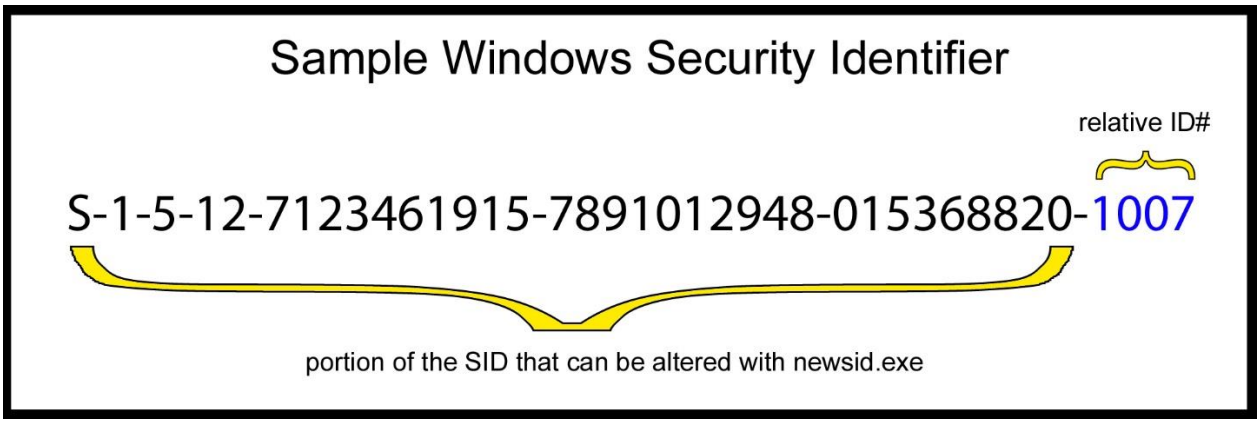
Sample Windows Security Identifier
relative ID#
S-1-5-12-7123461915-7891012948-015368820-1007
portion of the SID that can be altered with newsid.exe

Figure 3: Later, the SID becomes very important. The graphic above shows the format of Windows' Security Identifier (SID).

Unfortunately, you won't know if any of the listed characteristics have changed until you run FTK's decryption tool and find out.

These aren't normal circumstances:
In this particular investigation, the encrypted files were found on a drive (hereinafter called *suspect drive*) that was completely separate from the drive that had the Windows installation, which originally encrypted the files (*source drive)*. In addition, both drives were loose, found in a closet at the suspect's residence, and neither was enclosed within a functional computer system.

Essential Problem:
Encrypted files believed to contain direct evidence of sexual abuse are inaccessible and the traditional methods of decrypting those files are unavailable because no original machine is present to boot the source drive used to encrypt the files. The encrypted files are on a different volume than the original credentials so FTK is ineffective.

The following are some (but not all) of the decryption methods that were tried, and all of these attempts failed. The scope and details of every attempt to decrypt these files would be too detailed to list, so I'll stick to the high points:

Booting the source drive directly:
Attempts to clone the source drive failed using the Tableau TD-2 duplicator we have in the lab. It reported that there were sector errors and terminated the copy each time, with no option to bypass or accept the errors. Clearly this drive had some significant damage. The original plan was to copy the EFS documents onto a cloned version of the source drive, then import the whole drive into FTK to use FTK's EFS decryption. With no cloned drive, the only option would be to copy the encrypted files onto the original source drive, which is evidence, and I was unwilling to alter original evidence and thereby violate a basic rule of computer forensics.

PRTK
Direct use of Password Recovery Toolkit by importing a single encrypted file was an utter failure in every regard. It wouldn't even recognize that the files were encrypted with EFS, but instead tried to attack the files as if they were Truecrypt, BestCrypt, and a number of other encryption methods.

FTK
Importing both the image for the suspect drive with the encrypted files, as well as the source drive with the credentials seemed like a good idea, but FTK obviously does not look across multiple drives for EFS credentials because it

failed. I tried multiple variations of this, including the mounting of writeable volumes with FTK Imager, then importing credential files from the master account onto the suspect drive with the encrypted files, copying the encrypted files onto the original source drive then re-importing to FTK, etc. Nothing worked.

Importing credentials:
Directly importing the various EFS-related credentials from the source drive onto the forensic workstation failed, as expected. Only use of a recovery certificate generated by a live boot of the source drive will transfer decryption rights.

Booting the original image:
Since we didn't have the original machine to boot, I tried using a series of virtual machines, including LiveView, VmWare Workstation, and VirtualBox to boot the image captured from the source drive. Nothing worked, and even use of LiveView was an utter failure. LiveView is great at setting up VM configurations to bypass many of the boot errors one encounters when booting an image without the original hardware but it didn't come through on this, despite attempts on both Windows 7 and Windows XP workstations using LiveView .7 and .8b under both VmWare Workstation 10 and VmWare Server 1.8.

Here is what worked:
Since the research clearly shows that the only two ways to defeat EFS encryption are by a) having a recovery certificate generated by the original Windows User or b) opening the files on the original computer, I decided to "simulate" scenario b). I would recreate the original Windows User scenario by using a spare workstation and building a Windows install from scratch, then spoofing the necessary identification information using a variety of techniques.

Steps taken:

1. I took an old forensic workstation that had been retired from use because it was old and slow. I installed a new hard drive and completed a fresh install of Windows XP SP2, the identical operating system found on the original source drive.

2. Mounting the suspect drive as an image allowed viewing of the encrypted files' properties, including the name of the Windows user that encrypted the file as well as the thumbprint of the certificate information. Using that information, I gathered all the credential information from the source drive's user account, which were stored in the following paths:
    - Root/Documents and Settings/[user]/Application Data/Microsoft/Credentials
    - Root/Documents and Settings/[user]/Application Data/Microsoft/Crypto
    - Root/Documents and Settings/[user]/Application Data/Microsoft/Protect
    - Root/Documents and Settings/[user]/Application Data/Microsoft/System Certificates

3. Because the SID, Windows logon password, and Windows account name all needed to be identical to the original machine, the following steps were taken:
    a. Distributed Network Attack (a variant of Access Data's Password Recovery Toolkit using multiple machines across a network) was used to defeat all the user account passwords.
    b. A new account was created with the same user name as the

original, with the same password attached.

c. Because the SID was wildly different from the original, I used the program newsid.exe (free download) to change the SID to the correct number.

4. Unfortunately, despite the SID having the correct domain identifier, the relative ID # portion (at the very end) was wrong. Instead of a suffix of 1005, my newly-duped account SID had a suffix of 1007. Because of this, I was going to have to recreate the entire user account, making sure that the relative ID# was correct.

Some research online helped me discover that when creating a new user, Windows consults a registry key located at:

hklm\sam\sam\domains\account\f
(See Figure 4)

At location 0048, there is a value you can set to direct Windows what # the next user account created will have for its relative ID # - the number at the very end of the SID that I got wrong. It must be entered in hexadecimal code, and then the bytes need to be reversed for entry.1005 translated to hex is 03 ED, reversed is ED 03. Using *regedit*, a windows application, I entered "ED 03" at location 48 in the registry key, and then created a new user account using the appropriate control panel. Just as planned, it created the proper SID, ending in 1005, as hoped.

I then deleted the other user account ending in 1007, imported all my credentials into the new

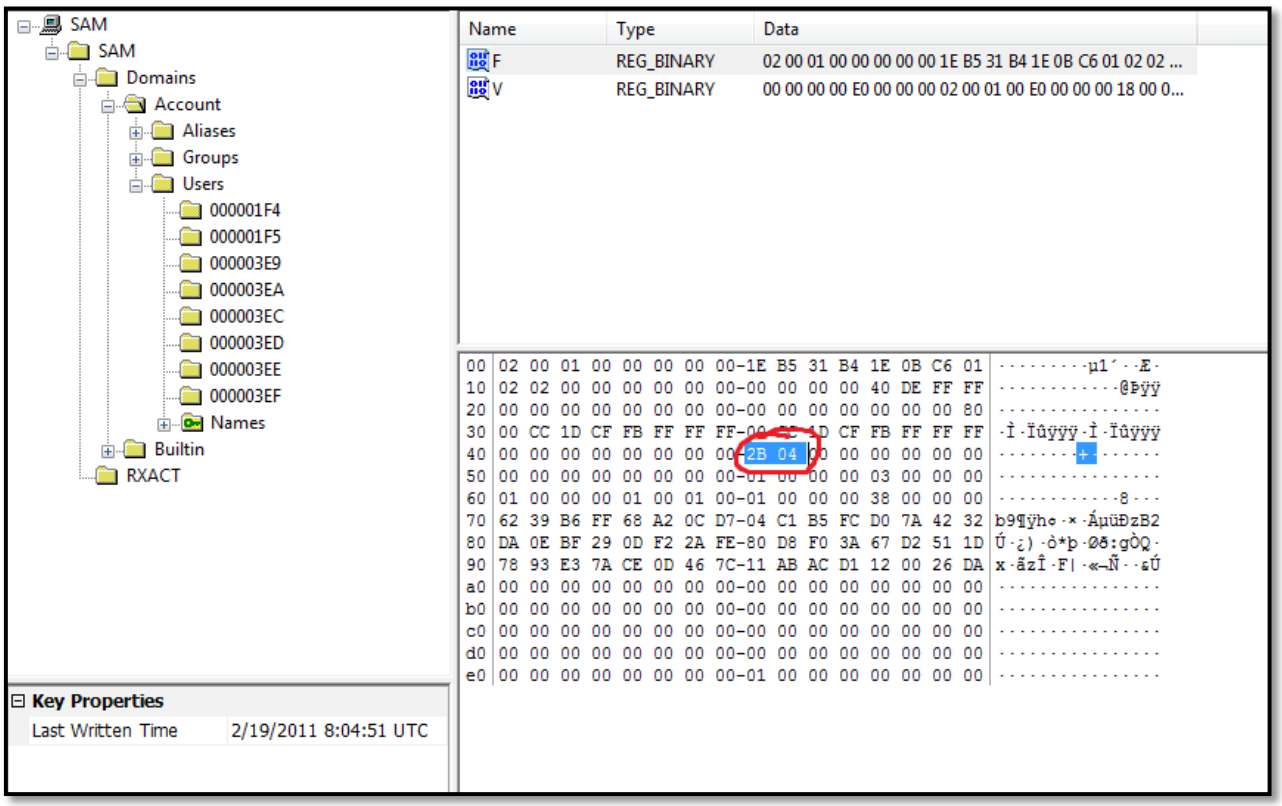

Figure 4: As illustrated above, Windows consults a registry key when creating a new user.

account with relative ID# 1005, and restarted the machine.

I attached a cloned version of my suspect drive via an external USB dock, entered in the Windows logon for my newly duped account, and all of the encrypted files were instantly visible. Success! I then exported a recovery certificate, which imported easily into one of our faster Windows 7 workstations for direct access to the files.

Finding the Motherlode:

Review of the now-decrypted files not only revealed over 8,000 images/videos of child erotica and child pornography, but the folder with our victim's name on it contained many videos of her abuse by our suspect; in some videos, the suspect clearly showed his face during the recorded abuse.

Using a metadata reader, the exact date/time of the offenses were documented for prosecution purposes. The abuse dates/times correlated with the time period where the suspect had unfettered access to the victim.

Acknowledgement:

A primer on EFS data recovery found at beginningtoseethelight.org was absolutely essential to success in this case. The author, clark@hushmail.com also provides strategies for recovering EFS encrypted files even if the original credentials or source disk aren't available.

http://www.beginningtoseethelight.org/efsrecovery/index.htm

About the author:

Sergeant David Willson is with the Alaska Bureau for Investigation, working in the Technical/Financial Crimes Unit. He can be reached at david.willson@alaska.gov.